Formal Methods

Assignment-2

Ali Sabeh, ID :201822768

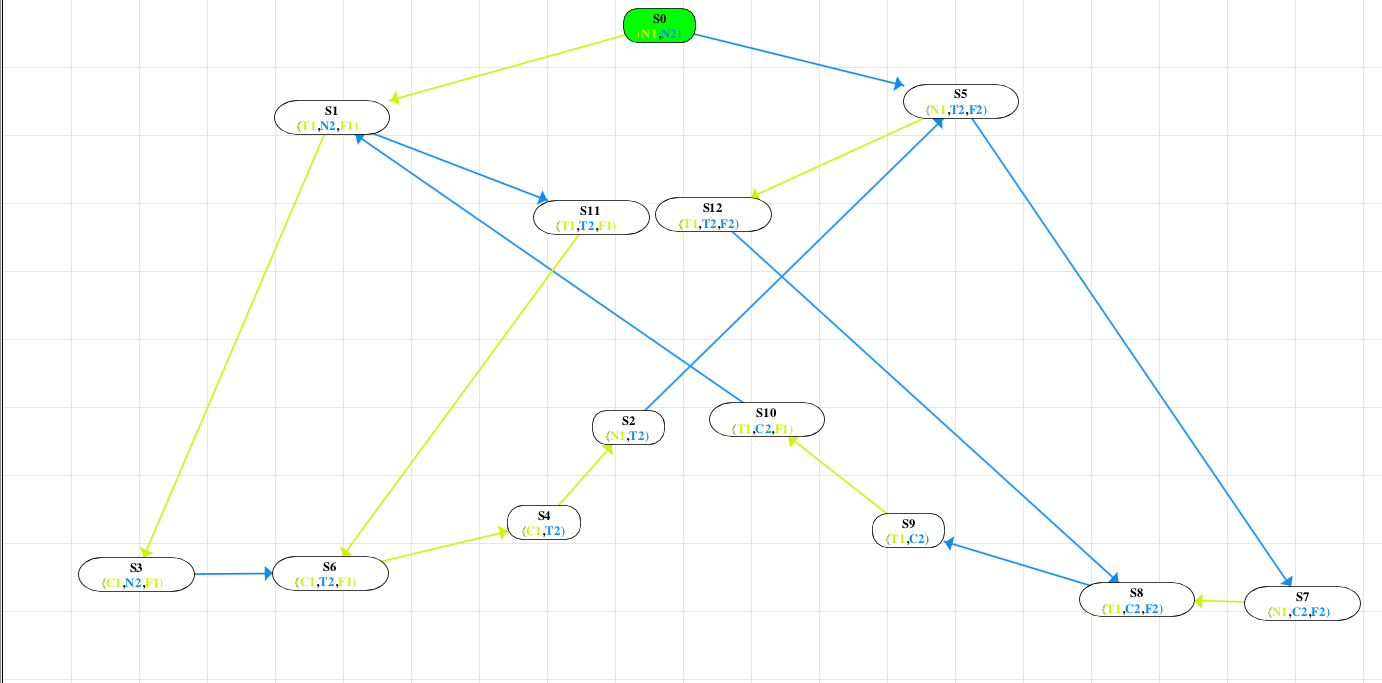
Exercise 1

CTL formula:

(AG( T1 => ( AF( C1 ) ) ) ) & ( AG( T2 => ( AF( C2 ) ) ) ) & ( EG( N1 ) ) & ( EG( N2 ) )

In the first Assignment we built a graph ( set of states which represents the process’s states and their transitions) , where this graph satisfies the following formula( graph is live and correct) :

(AG( T1 => ( AF( C1 ) ) ) ) & ( AG( T2 => ( AF( C2 ) ) ) ).

And the resulting graph for assignment 1 was : 

We need to modify this graph so it also satisfies : ( EG( N1 ) ) & ( EG( N2 ) ) , which means that there exists a path were N1 globally holds, and other path were N2 globally holds.

We can notice that if we follow the path S0 -> S1 -> S3, N2 is always true but moving forward to S6, we see that N2 is no more true. So we simply add new transition from S3 (C1,N2,F1 ) to S0 (N1,N2).

Same thing is applied to N1, we can notice that it holds in states S0 -> S5 -> S7, so from state S7 we add a transition to S0 (N1,N2 ) .

Another issue modified on the graph of assignment 1, is the 2 states S01(N1,N2,F1) and S02(N1,N2,F2) which are the result of S0(N1,N2), the result behind this change was to introduce an intermediate state so that we don’t have to modify and add states at the same time.

Therefore, we now have 2 paths (loops) , the first 1 ensures that N1 always holds, and the second path ensures that N2 always hold.

Using eshmun we constructed the below graph and check its correctness (model check). Moreover, using global model check we inserted the formula (EG( N1 ) ) & ( EG( N2 ) ) , and it shaded state S0(N1,N2).

**Briefly**

For process P1 to enter the critical section:

We start at process S0, give process P1 the mutex F1 so that we are now in state S01, then S1(T1,N2,F1) were process T1 requesting, then S3 were process p1 executing the critical section, then S6 were P2 requesting, then we set the mutex F1 to false in S4 and give process p2 the mutex F2 in S2, so that process P2 repeats the same algorithm as process P1. From states S3 and S7 were P1 and P2 are executing respectively we have two options either P1 remains executing/requesting( P2 remains neutral) or it alternates and let process p2 executes (In the first case). In the second case, were P2 can execute critical section, we can remain in a loop were P1 is neutral, or we can switch to P1 to execute.

In this case we ensured that process P1 or P2 can always execute, or they can alternate between each others.

The resulting Graph is as follows:

